Why is space not being freed from disk after deleting a file in Red Hat Enterprise Linux?

Solution Verified - Updated February 13 2022 at 3:06 AM -
English

Environment

Red Hat Enterprise Linux (RHEL)

Issue

- Why is space not being freed from disk after deleting a file in Red Hat Enterprise Linux?
- When deleting a large file or files, the file is deleted successfully but the size of the filesystem does not reflect the change.
- I've deleted some files but the amount of free space on the filesystem has not changed.
- The OS was holding several very large log files open with some as large as ~30G. The file was previously deleted, but only stopping and restarting the jvm/java process released the disk space. The `lsof` command shows the following output before restarting the java process

  Raw

  ```
  COMMAND     PID      USER   FD      TYPE     DEVICE   SIZE/OFF       NODE
  NAME
  :
  java      49097    awdmw   77w      REG      253,6 33955068440    1283397
  /opt/jboss/jboss-eap-5/jboss-as/server/all/log/server.log (deleted)
  ```

- When you perform a df, the storage shows 90+% utilized, however, there is not really that much written to that space.

Resolution

Graceful shutdown of relevant process

First, obtain a list of deleted files which are still held open by applications:

Raw

```
$ lsof | egrep "deleted|COMMAND"
COMMAND       PID    TID TASKCMD     USER   FD  TYPE  DEVICE    SIZE/OFF
NODE NAME
ora         25575   8194 oracle    oracle   33   REG   65,65  4294983680
31014933 /oradata/DATAPRE/file.dbf (deleted)
```

Note: check either the filesystem path within NAME field or the device number under DEVICE to match the filesystem of interest.

The `lsof` output shows the process with pid `25575` has kept file `/oradata/DATAPRE/file.dbf` open with file descriptor (fd) number `33`.

After a file has been identified, free the file used space by shutting down the affected process. If a graceful shutdown does not work, then issue the `kill` command to forcefully stop it by referencing the `PID`.

Truncate File Size

Alternatively, it is possible to force the system to de-allocate the space consumed by an in-use file by forcing the system to truncate the file via the `proc` file system. This is an advanced technique and should only be carried out when the administrator is certain that this will cause no adverse effects to running processes. Applications may not be designed to deal elegantly with this situation and may produce inconsistent or undefined behavior when files that are in use are abruptly truncated in this manner.

Raw

```
$ echo > /proc/pid/fd/fd_number
```

For example, from the `lsof` output above:

Raw

```
$ file /proc/25575/fd/33
/proc/25575/fd/33: broken symbolic link to `/oradata/DATAPRE/file.dbf (deleted)'
$ echo > /proc/25575/fd/33
```

The same reason will cause different disk usage from `du` command and `df` command, please refer to Why does df show bigger disk usage than du?

To identify the used file size (in blocks), use the command below:

Raw

```
# lsof -Fn -Fs |grep -B1 -i deleted | grep ^s  | cut -c 2- | awk '{s+=$1} END
{print s}'
```

Root Cause

On Linux or Unix systems, deleting a file via `rm` or through a file manager application will *unlink* the file from the file system's directory structure; however, if the file is still open (in use by a running process) it will still be accessible to this process and will continue to occupy space on disk. Therefore such processes may need to be restarted before that file's space will be cleared up on the filesystem.

Diagnostic Steps

Log Reaper will allow you to visualize and quickly narrow down the `lsof` data to exactly the subset you want to see

This solution is part of Red Hat's fast-track publication program, providing a huge library of solutions that Red Hat engineers have created while supporting our customers. To give you the knowledge you need the instant it becomes available, these articles may be presented in a raw and unedited form.

7 Comments

Log in to comment

Can I recover the deleted file, for example, save to another location?

Frequently coming on ext3 FS. Any way to mitigate it on Prod servers? Every time We can't restart app service?

this actually was very good, but in checking it out on an engineering system, there were plenty of 'deleted' files held by OS programs.
I would have concerns running systemctl restart on them and per your comment about knowing what we are doing emptying same on OS processes I would consider risky. firewalld and tuned for example.

gdm and Xorg files I was fine with, I have no problem kicking a few developers off restarting gdm, though it seems it didn't take long to add new deleted files after the restart.

i have accidentally deleted /var file system. how to recover without restoring.Please provide steps to recover.

RM Community Member 99 points
9 April 2019 8:08 AM
Ryszard Musielak
This is very helpful, however doesn't go far enough.

Raw

```
$ file /proc/25575/fd/33
/proc/25575/fd/33: broken symbolic link to
`/oradata/DATAPRE/file.dbf (deleted)'
```

Does this mean the process that is somehow "defunct"
and can be safely terminated using the kill command? It is
not usually possible to "shutdown the process gracefully", only a shutdown of the whole service like a database or application server can resolve the problem,

At some point I had such a huge buildup of these that we were running out of space on the / file system and the only option was to reboot the whole server which freed 70% of space used.

It simply means the copy of /oradata/DATAPRE/file.dbf that process 25575 has open on file descriptor 33 was deleted while this process was running. Since the copy of that file was open at the time it was deleted from the filesystem directory hierarchy, the symbolic link is broken (doesn't point to a visible file within the filesystem) -- but it also means the space that that file is occupying on disk cannot be recovered/reclaimed aka the on-disk file is essentially in a 'busy' state and cannot be actually removed from the filesystem/disk until all processes that currently have it open, close their access to that instance of the file.

If you accumulating a large number of deleted file instances that are open by processes, typically this points to a poorly written program that "forgets" to close open files once they are done with them or application architectural issues such that multiple deleted instances of one or more files accumulate within the filesystem.

We can easily reproduce the above as shown below:

Raw

```
dd if=/dev/zero of=realfile bs=1M count=10              [A]

./openfile realfile &                       # open the file for append access,
then sleep
ps aux | grep openfile | grep -v grep
user     21200  0.0  0.0   4052   432 pts/19   S    15:45   0:00 ./openfile
realfile
file /proc/21200/fd/3
/proc/21200/fd/3: symbolic link to `/home/   user/realfile'                  <==
[1]

rm realfile

file /proc/21200/fd/3
/proc/21200/fd/3: broken symbolic link to `/home/   user/realfile (deleted)'  <==
[2]

dd if=/dev/zero of=realfile bs=1M count=20                       <== [3], [B]

file /proc/21200/fd/3
/proc/21200/fd/3: broken symbolic link to `/home/   user/realfile (deleted)'  <==
[4]

lsof | egrep 'COMMAND|realfile'
COMMAND     PID      USER   FD      TYPE             DEVICE    SIZE/OFF       NODE
NAME
openfile  21200      user    3w      REG              253,1    10485760   24821678
/home/user/realfile (deleted)  [A] [5]

./openfile realfile &
lsof | egrep 'COMMAND|realfile'
COMMAND     PID      USER   FD      TYPE             DEVICE    SIZE/OFF       NODE
NAME
openfile  21200      user    3w      REG              253,1    1048
5760   24821678 /home/user/realfile (deleted)  [A]
openfile  21225      user    3w      REG              253,1    20971520   24821486
/home/user/realfile            [B] [6]

rm realfile ; dd if=/dev/zero of=realfile bs=1M count=20 ; ./openfile realfile &
[C]
rm realfile ; dd if=/dev/zero of=realfile bs=1M count=20 ; ./openfile realfile &
[D]
rm realfile ; dd if=/dev/zero of=realfile bs=1M count=20 ; ./openfile realfile &
[E]

lsof | egrep 'COMMAND|realfile'
COMMAND     PID      USER   FD      TYPE             DEVICE    SIZE/OFF       NODE
NAME                                [7]
openfile  21200      user    3w      REG              253,1    10485760   24821678
/home/user/realfile (deleted)  [A]
openfile  21225      user    3w      REG              253,1    20971520   24821486
/home/user/realfile (deleted)  [B]
openfile  21231      user    3w      REG              253,1    20971520   24821586
/home/user/realfile (deleted)  [C]
openfile  21234      user    3w      REG              253,1    20971520   24821626
/home/user/realfile (deleted)  [D]
openfile  21237      user    3w      REG              253,1    20971520   24821671
/home/user/realfile            [E]
```

The above script creates the symptoms you noticed and asked about. The script creates a copy of realfile (referenced as copy [A]). That file is opened by openfile which goes to sleep after opening file file simulating a process that is continuing to use that file (or just forgot to close it after it was done with it).

[1] The 'file' command produces normal looking result. But then we remove the file. Actually the rm does an unlink(), that is it unlinks/removes the file entry from the filesystem directory. The file will be actually deleted and the on-disk space reclaimedat a later time... but only after the file's reference count goes to zero. Every process that has opened the file bumps its reference count by 1. At the point we issue the rm command, no other process from this point forward will be able to open realfile instance [A]... that instance of the file is no longer present within the directory hierarchy of the filesystem.

[2] The 'file' command after the file has been deleted shows the "broken symbolic link", the link to the file still exists within the process -- it can still access, read, write to the file --, but the file instance the open file reference pointed to is no longer present within the visible filesystem. Aka, the file 'realfile' is no longer present within the directory structure of the filesystem but is still present on-disk.

[3] Recreate 'realfile' won't change the broken link, because while process 21200 has realfile [A] open, the recreated file is realfile [B] -- a file of the same name but one that occupies its own and different space within the filesystem/disk than instance [A].

[4] Since 'realfile' instance [A] is still not within the visible filesystem hierarchy.... so same broken symbolic link message.

[5] We see the deleted file within the lsof output. The file is still occupying disk space, but has been deleted from the filesystem visible directory hierarchy. The instance of realfile [A] is associated with inode number '24821678' (under NODE column).

[6] So we opened realfile [B] instance and another lsof shows that the two instances of the file are associated with two different NODE numbers -- that is they are two different files. The first one listed ([A] instance of the file), has been deleted from the directory hierarchy of the filesystem but because it is still open/referenced by a process, the space it occupies cannot be recovered/ reclaimed until such time as the file is closed by all processes accessing it.

[7] We repeat the process several times -- creating, opening, deleting an instance of the file -- which results in 4 instances A-D being no longer present within the file directory hierarchy (aka a user cannot see those files from the command line level), but processes are holding copies of those files open and all that space as indicated by SIZE column cannot be reclaimed until the processes holding those file open, either close the file or exit.

Within the above example there is 70MB of space that is being prevented from being reclaimed because the application 'openfile' has not closed access to the file.

I have a similar issue. I didn't delete an open file, I moved and open file. The lsof utility reports it as open but deleted. Since I have the file, what is the cleanest way to fix this? I want the file eventually moved to the new location where it is now, but I also need to do this cleanly and reclaim the space.